Notice of Allowability	Application No. 10/667,812	Applicant(s) NIKITIN ET AL.	
	Examiner Jason Mitchell	Art Unit 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--
All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *an application filed 9/22/03*.

2. ☒ The allowed claim(s) is/are *17-19, 22-27 (renumbered 1-9)*.

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)
1. ☒ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date *20070320* .
7. ☒ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

MENG-AI T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER

EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Mr. Chris Maiorana (reg# 42829) on March 20, 2007.

2. **The application has been amended as follows:**

3. **The Title is changed to:**

"METHOD FOR OPTIMIZING EXECUTION TIME OF PARALLEL PROCESSOR PROGRAMS"

4. **The claims are amended as follows:**

1-16. (CANCELLED).

17. (CURRENTLY AMENDED) A method for optimizing execution time of a parallel processor program, comprising:

(a) receiving said parallel processor program;

(b) performing dummy jumps optimization;

(c) performing linear code optimization, wherein for a current subcommand included in a current command in a current domain in said parallel processor program said linear code optimization further comprises :

1) evaluating a first list comprising objects that are used in an exclusive mode by the current subcommand and a second list comprising objects that are used by the current subcommand, but not changed by the current subcommand,

2) determining a first maximal address that is less than the address of the current command and greater than or equal to an address of a first command of the current domain such that a second command with said first maximal address includes a second subcommand, wherein a third list, comprising objects that are used in an exclusive mode by the second subcommand, intersects with said first list comprising objects that are used in an exclusive mode by the current subcommand, and assigning a value representing a null address to said first maximal address when no first maximal address is found,

3) determining a second maximal address that is less than the address of the current command and greater than or equal to an address of the first command of the current domain such that a third command with said second maximal address includes a third subcommand, wherein a fourth list, comprising objects that are used by the third subcommand, but not changed by the third subcommand, intersects with said first list comprising objects that are used in an exclusive mode by the current subcommand, and assigning said value representing said null address to said second maximal address when no second maximal address is found,

4) determining a third maximal address that is less than the address of the current command and greater than or equal to an address of the first command of the current domain such that a fourth command with said third maximal address includes a fourth subcommand, wherein a fifth list, comprising objects that are used in an exclusive mode by the fourth subcommand, intersects with said second list

comprising objects that are used by the current subcommand, but not changed by the current subcommand, and assigning said value representing said null address to said third maximal address when no third maximal address is found, and

5) defining a fourth maximal address as a maximum value of the group consisting of (i) said address of said first command of the current domain, (ii) said first maximal address incremented by one, (iii) said second maximal address and (iv) said third maximal address incremented by one, where said fourth maximal address is less than the address of the current command and greater than or equal to the address of the first command of the current domain;

(d) performing jumps optimization;

(e) returning back to said step (b) when there is any change made in said steps (b), (c), and (d); and

(f) outputting a resulted new processor program when there is no change made in said steps (b), (c), and (d).

18. (CURRENTLY AMENDED) The method of claim 17, wherein said step (b) ~~comprising~~ further comprises:

(b1) applying transition of jumps;

(b2) removing unreachable jumps; and

(b3) removing dummy jumps.

19. (CURRENTLY AMENDED) The method of claim 17, wherein said step (c) ~~comprising~~ further comprises:

~~(cl)~~ examining all commands included in ~~a~~ said current domain ~~<domain>~~; and

~~(c2)~~ removing empty commands from said current domain ~~<domain>~~.

20-21. (CANCELLED).

22. (CURRENTLY AMENDED) A computer-readable medium having computer-executable instructions for performing a method for optimizing execution time of a parallel processor program, said method comprising:

(a) receiving said parallel processor program;

(b) performing dummy jumps optimization;

(c) performing linear code optimization, wherein for a current subcommand included in a current command in a current domain in said parallel processor program said linear code optimization further comprises :

1) evaluating a first list comprising objects that are used in an exclusive mode by the current subcommand and a second list comprising objects that are used by the current subcommand, but not changed by the current subcommand,

2) determining a first maximal address that is less than the address of the current command and greater than or equal to an address of a first command of the current domain such that a second command with said first maximal address includes a second subcommand, wherein a third list, comprising objects that are used in an exclusive mode by the second subcommand, intersects with said first list comprising objects that are used in an exclusive mode by the current subcommand, and

assigning a value representing a null address to said first maximal address when no first maximal address is found,

3) determining a second maximal address that is less than the address of the current command and greater than or equal to an address of the first command of the current domain such that a third command with said second maximal address includes a third subcommand, wherein a fourth list, comprising objects that are used by the third subcommand, but not changed by the third subcommand, intersects with said first list comprising objects that are used in an exclusive mode by the current subcommand, and assigning said value representing said null address to said second maximal address when no second maximal address is found,

4) determining a third maximal address that is less than the address of the current command and greater than or equal to an address of the first command of the current domain such that a fourth command with said third maximal address includes a fourth subcommand, wherein a fifth list, comprising objects that are used in an exclusive mode by the fourth subcommand, intersects with said second list comprising objects that are used by the current subcommand, but not changed by the current subcommand, and assigning said value representing said null address to said third maximal address when no third maximal address is found, and

5) defining a fourth maximal address as a maximum value of the group consisting of (i) said address of said first command of the current domain, (ii) said first maximal address incremented by one, (iii) said second maximal address and (iv) said third maximal address incremented by one, where said fourth maximal address is less than the address of the current command and greater than or equal to the address of the first command of the current domain;

(d) performing jumps optimization;

(e) returning back to said step (b) when there is any change made in said steps (b), (c), and (d); and

(f) outputting a resulted new processor program when there is no change made in said steps (b), (c), and (d).

23. (NEW) The method of claim 17, further comprising finishing processing of said current subcommand when said fourth maximal address equals the address of the current command.

24. (NEW) The method of claim 17, further comprising:

when a number of subcommands contained in a fifth command with said fourth maximal address is less than a predetermined maximum number of subcommands for one processor command, appending said current subcommand to the fifth command, removing said current subcommand from said current command, and finishing processing said current subcommand.

25. (NEW) The method of claim 24, further comprising:

when the number of subcommands contained in the fifth command with said fourth maximal address is not less than the predetermined maximum number of subcommands for one processor command, incrementing the fourth maximal address; and when said fourth maximal address equals the address of the current command finishing processing of said current subcommand, otherwise repeating the steps of claim 24.

26. (NEW) A method for optimizing execution time of a parallel processor program, comprising:

(A) receiving said parallel processor program;

(B) performing dummy jumps optimization;

(C) performing linear code optimization;

(D) performing jumps optimization, wherein for a current command having a previous command immediately preceding said current command, said jumps optimization comprises:

(1) when (i) said previous command has no jump-subcommand, (ii) said current command has at least one jump-subcommand and does not depend from said previous command, and (iii) a sum of a number of subcommands contained in said previous command plus a number of subcommands contained in said current command is less than or equal to a predetermined maximum number of sub-commands in a processor command, performing the steps of:

a) when said current command is not a jump-target command, replacing said previous command with a union of said previous command and said current command, and removing said current command; and

b) when said current command is a jump-target command, and said current command contains either a jump-subcommand of type JUMP or a jump-subcommand of type RETURN, replacing said previous command with a union of said previous command and said current command;

(2) when (i) said previous command has at least one jump-subcommand but has no jump-subcommands of types CALL, RETURN and JUMP, and (ii) said current command does not depend from said previous command, performing the steps of:

a) when said current command is not a jump-target command, and the sum of the number of subcommands in said previous command plus the number of subcommands in said current command is less than or equal to the predetermined maximum number of subcommands in a processor command, replacing said previous command with a union of said previous command and said current command, and removing said current command;

b) when (i) said current command is a jump-target command and contains either a jump-subcommand of type JUMP or a jump-subcommand of type RETURN, and (ii) the sum of the number of subcommands in said previous command plus the number of subcommands in said current command is less than or equal to the predetermined maximum number of subcommands in a processor command, replacing said previous command with a union of said previous command and said current command; and

c) when (i) said current command is a jump-target command, (ii) said step b) of said step (2) is not applicable, and (iii) the sum of the number of subcommands in said previous command plus the number of subcommands in said current command is less than the predetermined maximum number of subcommands in a processor command, replacing said previous command with a union of said previous command and said current command, and appending a subcommand of type JUMP, with an address argument set to an address of said current command plus one, to an end of said newly replaced previous command;

(3) when (i) said current command has either a jump-subcommand of type JUMP or a jump-subcommand of type CALL, (ii) a jump-target command of said jump-subcommand has no jump-subcommand and does not depend from said current command, and (iii) a sum of the number of subcommands in said current command plus a number of subcommands in said jump-target command is less than or equal to the predetermined maximum number of subcommands in a processor command, inserting all subcommands of said jump-target command into said current command at a location before said jump-subcommand, and increasing a jump-address in said jump-subcommand by one; and

(4) when (i) said current command has a jump-subcommand of type JUMP and (ii) a jump-target command of said jump-subcommand has at least one jump-subcommand and does not depend from said current command, performing the steps of:

a) when (i) said jump-target command has only one jump-subcommand of either type JUMP or type CALL and has no more jump-subcommands, and (ii) a sum of the number of subcommands in said current command plus a number of sub-commands in said jump-target command is less than or equal to the predetermined maximum number of subcommands in a processor command plus one, inserting all subcommands of said jump-target command into said current command before said jump-subcommand, and removing said jump-subcommand from said current command;

b) when (i) said jump-target command has at least one non-jump-subcommand of any type and at least one jump-subcommand having a type selected from the group consisting of LOOP_INC_NOLESS, LOOP_INC_NOMORE, LOOP_DEC_NOLESS, LOOP_DEC_NOMORE, ZERO_JUMP and NONZERO_JUMP, and (ii) a sum of the number of subcommands in said current command plus a number of sub-commands in said jump-target command is less than or equal to the predetermined maximum number of subcommands in a processor command, inserting all subcommands of said jump-target command into said current command at a location before said jump-subcommand, and replacing said jump-subcommand with a subcommand of type JUMP having an address argument set to an address of said jump-target command plus one;

(E) returning back to said step (B) when there is any change made in said steps (B), (C), and (D); and

(F) outputting a resulted new processor program when there is no change made in said steps (B), (C), and (D).

27. (NEW) A computer-readable medium having computer-executable instructions for performing a method for optimizing execution time of a parallel processor program, said method comprising:

(A) receiving said parallel processor program;

(B) performing dummy jumps optimization;

(C) performing linear code optimization;

(D) performing jumps optimization, wherein for a current command having a previous command immediately preceding said current command, said jumps optimization comprises:

(1) when (i) said previous command has no jump-subcommand, (ii) said current command has at least one jump-subcommand and does not depend from said previous command, and (iii) a sum of a number of subcommands contained in said previous command plus a number of subcommands contained in said current command is less than or equal to a predetermined maximum number of sub-commands in a processor command, performing the steps of:

a) when said current command is not a jump-target command, replacing said previous command with a union of said previous command and said current command, and removing said current command; and

b) when said current command is a jump-target command, and said current command contains either a jump-subcommand of type JUMP or a jump-subcommand of type RETURN, replacing said previous command with a union of said previous command and said current command;

(2) when (i) said previous command has at least one jump-subcommand but has no jump-subcommands of types CALL, RETURN and JUMP, and (ii) said current command does not depend from said previous command, performing the steps of:

a) when said current command is not a jump-target command, and the sum of the number of subcommands in said previous command plus the number of subcommands in said current command is less than or equal to the predetermined maximum number of subcommands in a processor command, replacing said previous command with a union of said previous command and said current command, and removing said current command;

b) when (i) said current command is a jump-target command and contains either a jump-subcommand of type JUMP or a jump-subcommand of type RETURN, and (ii) the sum of the number of subcommands in said previous command plus the number of subcommands in said current command is less than or equal to the predetermined maximum number of subcommands in a processor command, replacing said previous command with a union of said previous command and said current command; and

c) when (i) said current command is a jump-target command, (ii) said step b) of said step (2) is not applicable, and (iii) the sum of the number of subcommands in said previous command plus the number of subcommands in said current command is less than the predetermined maximum number of subcommands in a processor command, replacing said previous command with a union of said previous command and said current command, and appending a subcommand of type JUMP, with an address argument set to an address of said current command plus one, to an end of said newly replaced previous command;

(3) when (i) said current command has either a jump-subcommand of type JUMP or a jump-subcommand of type CALL, (ii) a jump-target command of said jump-subcommand has no jump-subcommand and does not depend from said current command, and (iii) a sum of the number of subcommands in said current command plus a number of subcommands in said jump-target command is less than or equal to the predetermined maximum number of subcommands in a processor command, inserting all subcommands of said jump-target command into said current command at a location

before said jump-subcommand, and increasing a jump-address in said jump-subcommand by one; and

(4) when (i) said current command has a jump-subcommand of type JUMP and (ii) a jump-target command of said jump-subcommand has at least one jump-subcommand and does not depend from said current command, performing the steps of:

a) when (i) said jump-target command has only one jump-subcommand of either type JUMP or type CALL and has no more jump-subcommands, and (ii) a sum of the number of subcommands in said current command plus a number of sub-commands in said jump-target command is less than or equal to the predetermined maximum number of subcommands in a processor command plus one, inserting all subcommands of said jump-target command into said current command before said jump-subcommand, and removing said jump-subcommand from said current command;

b) when (i) said jump-target command has at least one non-jump-subcommand of any type and at least one jump-subcommand having a type selected from the group consisting of LOOP_INC_NOLESS, LOOP_INC_NOMORE, LOOP_DEC_NOLESS, LOOP_DEC_NOMORE, ZERO_JUMP and NONZERO_JUMP, and (ii) a sum of the number of subcommands in said current command plus a number of sub-commands in said jump-target command is less than or equal to the predetermined maximum number of subcommands in a processor command, inserting all subcommands of said jump-target command into said current command at a location before said jump-subcommand, and replacing said jump-subcommand with a subcommand of type JUMP having an address argument set to an address of said jump-target command plus one;

(E) returning back to said step (B) when there is any change made in said steps (B), (C), and (D); and

(F) outputting a resulted new processor program when there is no change made in said steps (B), (C), and (D).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Mitchell whose telephone number is (571) 272-3728. The examiner can normally be reached on Monday-Thursday and alternate Fridays 7:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

[illegible]
Jason Mitchell
3/21/07

[illegible]
MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 21